PROBABLE DIAGNOSTIC SYSTEM

AO

Patent number: JP2001318804
Publication date: 2001-11-16
Inventor: BARFORD LEE A; SMITH DAVID R
Applicant: AGILENT TECHNOL INC
Classification:
- international: G06F11/28; G06F11/22; G06F11/26; G06F17/18
- european:
Application number: JP20010083806 20010322
Priority number(s):

Also published as:

- EP1136912 (A2)
- US6691249 (B1) AM
- EP1136912 (A3)
- EP1136912 (B1)

Report a data error here

Abstract of **JP2001318804**

PROBLEM TO BE SOLVED: To provide a probable diagnostic engine for an assembly and for remote applicable task.
SOLUTION: The diagnostic engine for a device equipped with plural components receives a model for applying the test result of the test of one group related with the device when at least one test fails and the level of comprehension of the test to the components of the device and information describing probable dependency between the tests, and the diagnostic engine is provided with a means for setting or designating the number N of the components which are likely to simultaneously become defective and a calculating means for calculating the likelihood of each component constituting a sub-set equipped with the size of not more than N to be any defective component. Thus, it is possible to provide the diagnostic engine in which the calculation is substantially accurate within a floating point calculation error range.

PATENT ABSTRACTS OF JAPAN

AO

(11)Publication number : 2001-318804
(43)Date of publication of application : 16.11.2001

(51)Int.Cl. G06F 11/28
G06F 11/22
G06F 11/26
G06F 17/18

(21)Application number : 2001-083806
(22)Date of filing : 22.03.2001

(71)Applicant : AGILENT TECHNOL INC
(72)Inventor : BARFORD LEE A
SMITH DAVID R

(30)Priority
Priority number : 2000 533115 Priority date : 22.03.2000 Priority country : US

(54) PROBABLE DIAGNOSTIC SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a probable diagnostic engine for an assembly and for remote applicable task.
SOLUTION: The diagnostic engine for a device equipped with plural components receives a model for applying the test result of the test of one group related with the device when at least one test fails and the level of comprehension of the test to the components of the device and information describing probable dependency between the tests, and the diagnostic engine is provided with a means for setting or designating the number N of the components which are likely to simultaneously become defective and a calculating means for calculating the likelihood of each component constituting a sub-set equipped with the size of not more than N to be any defective component. Thus, it is possible to provide the diagnostic engine in which the calculation is substantially accurate within a floating point calculation error range.

LEGAL STATUS
[Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

AO

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－318804
（P2001－318804A）
(43)公開日　平成13年11月16日(2001.11.16)

(51)Int.Cl.[7]		識別記号	FI		テーマコード(参考)
G06F	11/28	340	G06F	11/28	340A
	11/22	360		11/22	360C
	11/26			11/26	
	17/18			17/18	Z

審査請求　未請求　請求項の数14　ＯＬ　（全 16 頁）

(21)出願番号　特願2001－83806(P2001－83806)
(22)出願日　平成13年３月22日(2001.3.22)
(31)優先権主張番号　０９／５３３１１５
(32)優先日　平成12年３月22日(2000.3.22)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　399117121
アジレント・テクノロジーズ・インク
ＡＧＩＬＥＮＴ　ＴＥＣＨＮＯＬＯＧＩＥＳ，　ＩＮＣ．
アメリカ合衆国カリフォルニア州パロアルト　ページ・ミル・ロード　395
(72)発明者　リー・エイ・パーフォード
アメリカ合衆国カリフォルニア州サンノゼ，ミットン・ドライブ　2931
(74)代理人　100099623
弁理士　奥山　尚一　（外２名）

最終頁に続く

(54)【発明の名称】　確率的な診断システム

(57)【要約】
【課題】　組み込み及び遠隔適用業務用の確率的な診断エンジンを提供する。
【解決手段】　複数のコンポーネントを備えた装置の診断エンジンであって、前記診断エンジンが、少なくとも１つのテストが不合格になった前記装置に関する１集合のテストのテスト結果と、前記装置のコンポーネントに対するテストの網羅度と、前記テスト間における確率的な従属性を記述する情報とを与えるモデルとを受信し、前記診断エンジンは、同時に不良になる可能性のあるコンポーネントの数Ｎを設定または指定するための手段と、Ｎ以下のサイズを備えた部分集合をなす前記コンポーネントのそれぞれが、不良コンポーネントである尤度を計算するための計算手段を含み、これにより、前記計算が浮動小数点計算誤差範囲内において実質的に正確であることを特徴とする診断エンジンを提供する。

1

【特許請求の範囲】

【請求項１】　複数のコンポーネントを備えた装置を診断するための診断エンジンであって、

前記コンポーネントの状態が、確率的に独立しているものと仮定して、任意の特定の集合をなす前記コンポーネントが不良であり、他の全てのコンポーネントが良好である確率を計算することと、

コンポーネントの状態が与えられると、いくつかのコンポーネントの機能性を同様にテストするために適用可能な共用関数の状態が、確率的に独立しているものと仮定して、ある特定の集合をなすコンポーネントが不良の場合に、任意の特定の集合をなす共用関数が不合格になり、別の特定の集合をなす共用関数が合格する確率を計算することと、

コンポーネント及び共用関数の状態が与えられると、前記装置に適用可能なテストの状態が、確率的に独立しているものと仮定して、ある特定の集合をなすコンポーネントが不良であり、残りのコンポーネントが良好であり、特定の集合をなす共用関数が不合格になり、残りの共用関数が合格になる場合には、任意の特定の集合をなすテストが不合格になり、別の特定の集合をなす共用関数が合格する確率を計算することとを含んでなり、前記診断エンジンが、

少なくとも１つのテストが不合格になった前記装置に関するある集合のテストのテスト結果と、

前記装置のコンポーネントに対するテストの網羅度と、前記テスト間における確率的従属性を記述する情報とを与えるモデルとを受信し、前記診断エンジンは、

同時に不良になる可能性のあるコンポーネントの数Ｎを設定または指定するための手段と、

Ｎ以下のサイズを備えた部分集合をなす前記コンポーネントのそれぞれが、不良コンポンーネントである尤度を計算するための計算手段とを含んでおり、これにより、前記計算が浮動小数点計算誤差範囲内において実質的に正確である診断エンジン。

【請求項２】　複数のコンポーネントを備えた装置を診断するための診断エンジンであって、

コンポーネントの状態が、確率的に独立しているものと仮定して、任意の特定の集合をなすコンポーネントが不良で、他の全てのコンポーネントが良好である確率を計算することと、

コンポーネントの状態が与えられると、いくつかのコンポーネントの機能性を同様にテストするために適用可能な共用関数の状態が、確率的に独立しているものと仮定して、ある特定の集合をなすコンポーネントが不良の場合、任意の特定の集合をなす共用関数が不合格になり、別の特定の集合をなす共用関数が合格する確率を計算することと、

コンポーネント及び共用関数の状態が与えられると、前記装置に適用可能なテストの状態が、確率的に独立しているものと仮定して、ある特定の集合をなすコンポーネントが不良で、残りのコンポーネントが良好であり、特定の集合をなす共用関数が不合格になり、残りの共用関数が合格する場合、任意の特定の集合をなすテストが不合格になり、別の特定の集合をなす共用関数が合格する確率を計算することとを含んでなり、

前記診断エンジンは、

前記コンポーネントの事前確率と、網羅度と、共用関数網羅度とを指定するための手段と、

合格または不合格になったテストあるいは実施されなかったテストがどれかを指定するための手段と、

同時に不良になる可能性のあるコンポーネント数Ｎを設定または指定するための手段と、

Ｎ以下のサイズを備えた部分集合をなすコンポーネントのそれぞれが、不良コンポンーネントである尤度を計算するための計算手段を含み、これにより、前記計算が浮動小数点計算誤差範囲内において実質的に正確であることとを含んでなる診断エンジン。

【請求項３】　１つ以上の前記コンポーネントに関して、前記コンポーネントが不良である確率を出力するための手段をさらに含んでいる請求項１または２に記載の診断エンジン。

【請求項４】　同時に不良になる可能性のあるコンポーネント数Ｎに関するデフォルト状態の値を設定するための手段をさらに含み、前記デフォルト状態の値が望ましくは１または２である請求項１または２に記載の診断エンジン。

【請求項５】　各コンポーネントの状態が、良好または不良であること、及び／または、各共用関数の状態が、合格または不合格であること、及び／または、各テストの状態が、合格または不合格である請求項１または２に記載の診断エンジン。

【請求項６】　診断中に、前記診断エンジンが必要とするメモリの量が、前記モデルの記憶に必要なメモリの量と、一定の因子を掛けて出力を記憶するのに必要なメモリの量より少ない請求項１または２に記載の診断エンジン。

【請求項７】　前記計算手段が、（数２４）の方程式から（数２８）の方程式を組み合わせた（数１４）の方程式に基づいて、前記不合格の確率を計算する請求項１または２に記載の診断エンジン。

【請求項８】　（数２４）の方程式における和を累算する浮動小数点レジスタまたは記憶場所と、

（数２５）の方程式における積を累算する浮動小数点レジスタまたは記憶場所と、

（数２７）の方程式における積を累算する浮動小数点レジスタまたは記憶場所と、

（数２８）の方程式における積を累算する浮動小数点レジスタまたは記憶場所とを含んでいる請求項７に記載の診断エンジン。

3

【請求項９】　暫定的な活動共用関数及び活動共用関数が望ましい共用関数を計算するための手段をさらに含んでいる請求項１または２に記載の診断エンジン。

【請求項１０】　前記共用関数を計算するための手段が、（数３３）の方程式に基づいて共用関数を計算している請求項９に記載の診断エンジン。

【請求項１１】　複数のコンポーネントを備えた装置を診断するための方法であって、

コンポーネントの状態が、確率的に独立しているものとものと仮定して、任意の特定の集合をなすコンポーネントが不良で、他の全てのコンポーネントが良好である確率が計算されるステップと、

コンポーネントの状態が与えられると、いくつかのコンポーネントの機能性を同様にテストするために適用可能な共用関数の状態が、確率的に独立しているものと仮定して、ある特定の集合をなすコンポーネントが不良の場合、任意の特定の集合をなす共用関数が不合格になり、別の特定の集合をなす共用関数が合格する確率が計算されるステップと、

コンポーネント及び共用関数の状態が与えられると、前記装置に適用可能なテストの状態が、確率的に独立しているものと仮定して、ある特定の集合をなすコンポーネントが不良で、残りのコンポーネントが良好であり、特定の集合をなす共用関数が不合格になり、残りの共用関数が合格する場合、任意の特定の集合をなすテストが不合格になり、別の特定の集合をなす共用関数が合格する確率が計算されるステップとを含んでなり、前記方法が、

少なくとも１つのテストが不合格になった前記装置に関する１集合のテストのテスト結果を受信するステップと、

前記装置のコンポーネントに対するテストの網羅度と、前記テスト間における確率的従属性を記述する情報とを与えるモデルを受信するステップと、

同時に不良になる可能性のあるコンポーネントの数Ｎを設定または指定するステップと、不良コンポーネントである確率を計算するステップとを含んでなり、前記計算が浮動小数点計算誤差範囲内においてほぼ正確である診断方法。

【請求項１２】　コンピュータのようなデータ処理システムにて実行する場合に、請求項１０に記載の方法を実行するためのデータ記憶媒体に記憶されるのが望ましいソフトウェア製品。

【請求項１３】　任意の適合するデータ処理装置において、または、前記処理装置によって実行する場合に、請求項１０に記載の方法のステップを実行するための任意の種類のデータ記憶媒体に記憶されるか、または、前記データ記憶媒体によって別様に供給されるソフトウェア・プログラム。

【請求項１４】　複数のコンポーネントを備えた装置を診断するための診断エンジンであって、

コンポーネントの状態が、確率的に独立しているものと仮定して、任意の特定の集合をなすコンポーネントが不良で、他の全てのコンポーネントが良好である確率を計算する手段と、

コンポーネントの状態が与えられると、前記装置に適用可能なテストの状態が、確率的に独立しているものと仮定して、ある特定の集合をなすコンポーネントが不良で、残りのコンポーネントが良好である場合に、任意の特定の集合をなすテストが不合格になる確率を計算する手段とを含んでなり、前記診断エンジンが、

少なくとも１つのテストが不合格になった、前記装置に関する１集合のテストのテスト結果と、

前記装置のコンポーネントに対するテストの網羅度、及び、前記テスト間における確率的従属性を記述する情報を与えるモデルとを受信し、前記診断エンジンは、

同時に不良になる可能性のあるコンポーネントの数Ｎを設定または指定するための手段と、

Ｎ以下のサイズを備えた部分集合をなすコンポーネントのそれぞれが、不良コンポンーネントである確率を計算するための計算手段とを含み、前記計算が浮動小数点計算誤差範囲内において実質的に正確である診断エンジン。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、システムにおける故障のモニタと検出と分離とに関するものであり、特に、システムの分析に用いられるツールに関するものである。

【０００２】

【従来の技術】「診断」は、誤動作した装置が、誤った動作をしている理由を判定することを意味するものである。より改まった表現をすれば、診断とは、誤った動作を生じさせる所定の原因集合のうちの部分集合を選択することである。診断では、誤った動作の説明と、正しさの確率または誤った診断のコストといった何らかの目的関数の最適化の両方を実施する必要性がある。この診断の必要性が、測定またはテストのための一般的な理由である。

【０００３】次に、修復またはプロセス改良のための特別に設計された装置の診断について考察することにする。これは、例えば、いつでも生成または破壊することが可能なソフトウェア・オブジェクトを含む分散コンピュータ・システムとはおおいに異なるものである。装置は、有限数の交換可能なコンポーネントから構成されているものと仮定する。装置の故障は、１つ以上の不良コンポーネントを備えることによってのみ発生する。本明細書において「診断」と呼ばれるものは、「故障識別（fault identification）」と呼ばれる場合が多い。故障装置が生じると、技術者またはコンピュータ・プログ

5

ラム（「テスト・エグゼクティブ（test executive）」と呼ばれる場合もある）は、１つ以上のテストを実行することになる。故障装置の内部動作に習熟した技術者は、テスト結果を解釈して、不良コンポーネントを識別しなければならない。

【０００４】例えば、“An expert system for diagnosing and maintaining the AT&T3B4000 computer: an architectural description"ACM, 1989においてJ.A. Kavicky及びG.D.Kraftによって解説されているように、コンピュータの故障診断にはエキスパート・システムが利用されてきた。Fitzgerald,G.L.の“Enhance computer fault isolation with A history memory”, IEEE, 1980には、オン・バス診断ハードウェアからのデータ解析に関する記載がある。冗長処理及び記憶素子と、データ・パスウェイと、故障装置をオフにして良好な冗長装置にスイッチする時機を決定する組み込みモニタ機能とを備えた故障許容コンピュータ（Fault-tolerant computer）が、何年にもわたって造られてきた（例えば、ＵＳ－Ａ－５，０９９，４８５号公報を参照されたい）。

【０００５】テストを受けるシステム（system under test：以下、「ＳＵＴ」と呼ぶ。）における故障の可能性のあるコンポーネントを判定するための従来の診断システムは、モデル・ベース診断システム（model-based diagnostic system）を含んでいる。モデル・ベース診断システムは、診断システムに対する入力として、用いられたテストからの実際のＳＵＴ応答、及び、正しいまたは間違ったＳＵＴ動作の対応するモデルを利用して、ＳＵＴの状態について結論を下す診断システムと定義することが可能である。こうした診断システムは、通常、ＳＵＴ及びそのコンポーネントのコンピュータ生成モデル、及び、診断プロセスに基づくものである。

【０００６】モデル・ベースの診断システムは、例えば、W. Hamscher, L. Console, J. deKleerの「Readings in system modek-based diagnosis」, Morgan Kauffman, 1992によって既知のところである。テスト・ベースのシステム・モデルは、Hewlett-Packard HP Fault Detective(ＨＰＦＤ)によって利用されており、HP FaultDetective User's Guide, Hewlett-Packard Co. , 1996に解説がある。

【０００７】ＵＳ－Ａ－５，８０８，９１９号公報（Preist他）は、モデル化の負担が大幅に軽減される機能テストに基づくモデル・ベースの診断システムを開示している。Preist他の特許において開示されたモデルは、機能テストのリスト、各機能テストによって用いられるコンポーネントのリスト並びに各コンポーネントが各機能テストによって用いられる程度、及び、個々のコンポーネントに関する履歴的なまたは推定的な事前故障率（a priori failure rate）を用いる。

【０００８】ＵＳ－Ａ－５，９２２，０７９号公報（Booth他）は、テスト・スーツ（testsuits：可能性のある故障を検出し、それらを弁別するモデルの能力）によって、可能性のある故障を識別し、間違った診断に基づく、起こりそうなモデル化エラーも識別する自動分析及び障害解決システムを開示している。

【０００９】ＥＰ－Ａ－８８７７３３号公報（Kanevsky他）は、テストを受ける装置の処理しやすいモデルに基づいて、まだ適用されていないテストの中から、テストを受ける装置に次に適用する１つ以上のテストを選択可能にする自動ツールをもたらすモデル・ベースの診断システムを開示している。

【００１０】上記３つのモデル・ベースの診断システムの場合には、診断エンジンは、診断に対するシステム・モデル・ベースのアプローチと確率的アプローチとを組み合わせている。診断エンジンは、テスト・スーツの結果を利用して、ＳＵＴのシステム・モデルに基づいて、最も故障の可能性が高いコンポーネントを計算する。

【００１１】診断エンジンは、故障した装置を、予め設計した１集合のテストからのテストを実施するように、所定の集合をなすテスト及び測定装置を用いてデバッグされることになるアプリケーションに利用することが可能である。診断エンジンは、ＳＵＴ及びＳＵＴのために決定されたシステム・モデルにより実行した実際のテストから得られたテスト結果を用いて、ＳＵＴのコンポーネントに関する故障候補のリストを計算する。例えば、テストが合格または不合格になると、コンポーネントの事前故障確率（a priori failure probability）から開始して、モデル情報によって、これらの確率に相応の重み付けをすることが可能である。少なくとも１つのテストが不合格にならなければならない。さもなければ、ＳＵＴは良好であると仮定される。

【００１２】組み込みプロセッサは、別の製品に組み込まれる（すなわち、内蔵される）ために計算力及び／またはメモリ・サイズの制限が厳しいマイクロプロセッサまたは他のデジタル計算回路である。組み込みプロセッサを含む製品の例は、一般に、自動車と、トラックと、主要な家庭用器具と、サーバ・クラス・コンピュータ（中央演算処理装置以外に、組み込み保守プロセッサを含む場合が多い）とを含んでいる。組み込みプロセッサは、一般に、デスクトップ・パーソナル・コンピュータに比べて、利用可能なメモリが数桁小さく、計算力は１～２桁低い。例えば、１メガバイトのメモリは、家庭用器具にとってはかなりの量になる。このような製品におけるこうした組み込みプロセッサが、製品の故障を診断可能にすることが望ましい。こうした能力をもたらす診断エンジンを、組み込み診断エンジンと呼ぶことにする。

【００１３】前述のHP Fault Detective製品またはＵＳ－Ａ－５，８０８，９１９号公報（Preist他）によって用いられているようなさまざまなヒューリスティック法（heuristic method）によって、確率的な診断を実施す

ることが可能である。ヒューリスティックは、本質的に、多少の正確度と計算時間の短縮とのトレード・オフを行う。しかし、HP Fault Detectiveは、一般に、４～８メガバイトのメモリを必要とする。これは、組み込み診断エンジンにとって法外な量になる可能性がある。

【００１４】この問題を解決するためのもう１つの方法は、モンテカルロ・シミュレーションである。モンテカルロ・シミュレーション法は任意の正確度にすることが可能であるが（シミュレーション数を増すことによって）、シミュレーション結果は、診断エンジンが後で読み取るデータ・ベースに記憶しなければならない。既述のように、空間効率のよい２進形式で記憶される場合でも、このデータ・ベースは、典型的なアプリケーションの場合、２～６メガバイトを必要とする。これは、組み込みアプリケーションにとってあまりにも多すぎ、診断毎に、データ・ベースをコンピュータ・ネットワークにアップロードしなければならない可能性のある分散アプリケーションに対して負担になる。

【００１５】確率的診断システムを構築する一般的な方法は、ベイズ・ネットワーク（Bayesian network）を利用することである（Finn V. Jensenの“Bayesian Networks”, Springer Verlag, 1997を参照されたい）。ベイズ・ネットワークは、閉路なし有向グラフ（directed acyclic graph：以下、「ＤＡＧ」と呼ぶ。）である。グラフの各ノードは、確率変数を表している。グラフのエッジは、２つの確率変数間における確率依存性を表している。ソース（エッジ内にないノード）は、他の全ての確率変数から独立しており、その事前確率がタグ付けされる。非ソース・ノードは、それが依存する全ての確率変数に左右されるそのノードの確率変数の値に対する確率を与えるテーブルによりタグ付けされる。

【００１６】ほとんどの診断用途のベイズ・ネットワークに基づく計算は、信念リビジョン（belief revision）と呼ばれる。確率変数のいくつかの値（この場合、いくつかのテストの結果に関する）が観測されるものと仮定する。観測された値が与えられると、信念リビジョン・アルゴリズムによって、観測されなかった全ての確率変数について、最も可能性の高い確率が計算される。信念リビジョンは、ＮＰ困難であり（M.r.Garey and D. S. Johnsonの“Computers and Intractability: A guide to the theory of NP-completeness”, W.H.Freeman and Co., 1979を参照されたい）、従って、既知の全てのアルゴリズムが、グラフのいくつかの確率変数に、指数関数的な最悪ケース計算時間を備えている。

【００１７】診断に用いられるベイズ・ネットワークは、確率変数と、テスト結果と診断を受ける装置及びそのコンポーネントの観測不能状態と故障仮説との間の任意の原因・結果関係を表すそれらの依存性とによって構成されている。グラフは、極めて大きく成長し、任意のトポロジを備えることが可能である。例えば、ヒューレット・パッカード社がプリンタの診断に用いた実験ベイズ・ネットワークは、２０００を超えるノードを備えている。こうしたネットワークの複雑性によって、下記の２つの問題点が生じる。

・非ソース・ノードの条件付き確率の全てを得るか、または、推定しなければならない。

・トポロジまたは条件付き確率に対する局所的変更によって、診断の正確度に理解し難い大局的影響が及ぶ可能性がある。

【００１８】換言すれば、任意のトポロジの大規模なベイズ・ネットワークを診断に用いることには、支援性の問題に関してルール・ベースの診断システムと多少とも同じ潜在的可能性がある。

【００１９】

【発明が解決しようとする課題】本発明の目的は、組み込み及び／または遠隔アプリケーションに用いることも可能な改良した確率診断を提供することにある。この目的は、独立クレームによって解決される。望ましい実施態様は、従属クレームによって示される。

【００２０】

【課題を解決するための手段】本発明によれば、例えば、最も故障した可能性の高いコンポーネントを識別することによって、デバッギング・プロセスの各段階において、技術者を自動的に支援するツールである診断エンジンが得られる。

【００２１】他の診断エンジンと比べた本発明の主たる利点は、メモリ・フットプリントを小さくすることができるという点であるため、コードと実行時の両方の必要メモリ量がわずかであり、モデル・サイズと線形に成長するだけである。

【００２２】本発明は、完全にＪａｖａで書くのが望ましく（例えば、James Gosling,BillJoy,and Guy Steel: The Java Language Specification, Addison Wesley,1996を参照されたい）、また、Ｊａｖａ標準言語ライブラリ・パッケージからの少数のクラスだけを用いるのが望ましい。これらの特徴によって、本発明は、組み込み及び分散アプリケーションに特に適したものになる。

【００２３】本発明は、故障装置が、あらかじめ設計した１集合のテストからのテストを実施する所定の集合をなすテスト及び測定装置を利用してデバッグされるアプリケーションにおいて用いられることを意図したものである。本明細書における目的からして、テストは、ある装置に対して実施されるプロシージャである。テストでは、可能性のある有限数の結果が生じる。多くのテストでは、２つの結果、すなわち、合格と不合格が生じる。例えば、コンピュータを修理するためのテストには、電源電圧が４．９～５．１ボルトの間であるか否かを確認する検査を必要とする可能性がある。もしそうであれば、テストは合格である。そうでなければ、テストは不合格である。テストには、故障モード（failure mode）

と呼ばれる付加的結果を生じる場合もある。例えば、あるテストでは、自動車の始動を試みることが必要とされる場合がある。自動車が始動すれば、テストは合格である。故障モードに含まれる可能性があるのは、次の通りである。

・キーを回すと、ライトが薄暗くなり、ボンネットの下からのノイズが生じない。

・キーを回すと、ライトは明るいままで、単一クリック・ノイズが生じる。

・キーを回すと、ライトは明るいままで、クリックが生じ、スタータのモータは回転するが、エンジンはかからない等。

【００２４】特定の装置をデバッグするために利用可能な集合をなす全テストは、その装置のテスト・スーツ（test suit）と呼ばれる。多くのアプリケーションは、デバッギング及びテストに関するこれらの定義に適合する。例としては、次のようなものがある。

・コンピュータ及び電子機器のサービス及び製造のやり直し。

・自動車及び家庭用器具のような製品のサービス。

・言語質問に対する返答を得ることを含むように「テスト」の概念を拡大する場合には、電話支援がこのモデルに適合する。

【００２５】・少なくとも１つのテストが不合格になった、物理的オブジェクトに対する１集合のテスト（例えば、テスト１＝合格、テスト２＝不合格、テスト３＝合格等）と、

・オブジェクトのコンポーネント（例えば、現場で交換可能な装置）に対するテストの網羅度、及び、テスト間の確率依存性を表した情報を提示するモデルとを与えると、本発明は、オブジェクトの確率的診断、すなわち、そのそれぞれの構成要素に、

・１つ以上のコンポーネントのリストと、

・それらのコンポーネントが不良コンポーネントである尤度または確率（尤度は非正規化確率である。すなわち、確率は、合計すると１にならなければならないが、尤度はその必要がない。）とを含んでいるリストを出力する。

【００２６】大部分の自動診断システムは、確率による重み付けを施さずに、可能性のある診断リストを提供するだけである。大部分の現場で交換可能な装置（field replaceable unit：以下、「ＦＲＵ」と呼ぶ。）は、ほとんどの場合、可能性があるという診断になることがわかるので、確率を示すことは、ＦＲＵの数が少ないアプリケーションにおいてとりわけ望ましい。確率によって、技術者は、技術者自身の専門的判断を適用する機会も与えられる。

【００２７】本発明によれば、複数のコンポーネント故障を取り扱うことが可能になる。単一故障と複数故障との区別は行われない。

【００２８】本発明は、診断に対するモデル・ベースのアプローチ（W.Hamscher,L.Console, and J.de Kleer: Readings in model-based diagnosis, Morgan Kauffman,1992）と確率的アプローチを組み合わせたものである。本発明は、前述のHP FaultDetectiveまたはＵＳ－Ａ－５，８０８，９１９号公報（Preist他）におけるものと同じテスト・ベース・モデルを用いている。このモデルは、テストと、テストを書いた技術者にとって便利であることを意図したやり方でテストされるコンポーネントとの確率関係を記述している。このモデルの特徴は、下記の通りであることが望ましいと思われる。

・２レベルの部分全体階層。すなわち、コンポーネント（現場で交換可能な装置）とそのサブコンポーネントの名前。

・コンポーネントの事前故障確率の推定値。

・テスト・スーツにおけるテスト名。

・各コンポーネントに関する各テストの網羅度、すなわち、テストによって働かされるコンポーネントの機能性の割合、あるいは、より明確に言えば、コンポーネントが不良の場合に、テストが不合格になる条件付き確率の推定値。

・全く同じ方法でいくつかのコンポーネント機能性をテストするので、依存しているテストをモデル化する方法であるテストの共用網羅度（例えば、共通ケーブルを介して所定のコンポーネントにアクセスする２つのテストは、そのケーブルに関する共用網羅度を備えている）。

・合格及び不合格以外に、テストに関する故障モードを指定する方法。故障モードは、名前と、コンポーネントまたはサブコンポーネントの２つのリストを備えている。解除リスト（acquit list）と呼ばれる第１のリストには、故障モードが発生するためには、何らかの操作可能な機能性を備えていなければならないコンポーネントまたはサブコンポーネントが挙げられている。非難リスト（indict list）と呼ばれる第２のリストには、故障モードが発生すると、不良の可能性のあるコンポーネントまたはサブコンポーネントが挙げられている。解除リスト及び非難リストの各項目は、故障モードによって働かされるコンポーネントの機能性の量の推定値も含んでいる。

【００２９】モデルは、例えば、下記のようにして生成することが可能である：

・例えば、前述のHP Fault Detectiveに付属しているモデル構築グラフィカル・ユーザ・インターフェイス（以下、「ＧＵＩ」と呼ぶ。）を利用して。HP FaultDetectiveモデルは、ＡＳＣＩＩファイルとして保管することが可能な、本発明によって内部的に用いられるより単純な形にそれを変換するプログラムによって読み取られる。本発明は、こうしたファイルを、ファイル・システム、ＵＲＬ、または、ローカル・メモリからロードすることが可能である。

・ＡＳＣＩＩテスト故障検出モデル（ASCII test Fault Detective Model）（．ｆｄｍ）ファイルを書くことによって、あるいは、
・Ｊａｖａによるモデル生成アプリケーション・プログラミング・インターフェイス（以下、「ＡＰＩ」と呼ぶ。）によって行う。

【００３０】特定のコンポーネントが不良であることが分かっている場合、このモデルは、数学的論理の規則と共に、あるテストが不合格になる確率の計算を可能にする。これらのモデル及びモデル構築プロセスに関するさらなる詳細については、同一出願人による同時係属の米国特許出願（出願人の国際参照番号：ＵＳ　２０－９９－００４２号公報）及びＵＳ－Ａ－５，９２２，０７９号公報（Booth他）に開示されている。モデル及びモデル構築プロセスの説明に関する前者の文書の教示は、引用することにより本明細書の一部をなすものとする。

【００３１】本発明によれば、良好または不良であることが分かっているコンポーネントのパターンが与えられると、テストの不合格の確率を計算することが可能になる。ベイズの定理（Bayes'Theorem）として知られる論理式によって、この計算を逆に実行することが可能になる。本発明では、特定のテスト結果が与えられると、ある特定のパターンをなすコンポーネント故障及び非故障の発生する確率を計算することが可能になる。次に、本発明では、コンポーネント故障／非故障の可能性のある全てのパターンを列挙し、テスト結果が与えられると、各パターンの確率を評価する。確率が最も高いパターンを診断として選択する。

【００３２】もちろん、１つのテストが、曖昧さのない診断を行うのに十分であることはほとんどない。そのテストがうまくいけば、いくつかのコンポーネントの疑いを晴らすことができるが、元凶を指摘することはできない。テストが不合格の場合、いくつかのコンポーネントを非難することが可能であり、コンポーネントの一部の疑いを晴らし、別のコンポーネントの一部に疑いの的を絞るため、他のテストが必要になる。（ここで、「疑いを晴らす」は、計算された故障確率を下げることを表し、「疑いの的を絞る」は、確率を最高値または最高値の近くまで上げることを表している。）テストが互いに独立している場合、複数のテスト結果の処理は、容易かつ迅速である。しかし、テストが独立していない場合には、問題ははるかに複雑になる。依存性は、共用関数（shared function）によってモデル化される。共用関数が合格または不合格になる可能性のある全ての様態、及び、それらがテスト結果の結合確率に及ぼす影響の程度について、個々別々に分類しなければならない。次に、例えば、下記に示すような診断アルゴリズム（擬似コードによる）の概要において略記されているように、これら全ての影響を合計しなければならない。

１．不良コンポーネントの可能性のある各組み合わせ毎に、
（ａ）合計を０にセットする。
（ｂ）共用関数の可能性のある合格／不合格の各組み合わせ毎に、
ｉ．観測されるテスト結果の確率を計算する。
ｉｉ．その確率を合計に加算する。
（ｃ）合計が与えられると、不良コンポーネントの組み合わせの尤度を計算する（ベイズの定理を利用して）。
２．故障尤度を降順にソートする。

【００３３】このアルゴリズムは、故障コンポーネントの組み合わせに対して反復され、合格及び不合格テストが与えられると、各組み合わせの条件付き尤度を計算する。

【００３４】この方法は、故障の全ての組み合わせに対する共用関数の結果の全ての組み合わせを調査するので、膨大な量の計算を要する可能性があるのは明らかである。

【００３５】この全てを実施すべき方法、さらには、この方法を実用的なものにすることができるように、計算負担を軽減する方法の数学的詳細については、「発明の実施の形態」のセクションにおいて詳細に示される。

【００３６】本発明によって用いられるモデルは、どれでも、ベイズ・ネットワークによって表すことが可能である。結果得られるグラフは、３つの部分であるソースと、シンクと、あるレベルの内部ノード（後で示される）とからのみ構成される。コンポーネント毎に１つのソースが存在する。テスト毎に１つのシンクが存在する。各共用関数は、１つの内部ノードによって表される。しかし、テスト網羅度情報を表すため、いわゆる「ノイジ・オア（Noisy-or）」（Finn V.Jensen：“Bayesian Networks”,Springer Verlag, 1997の第３章に定義され、詳述されている）構成を利用しなければならない。テスト網羅度情報の形式によって、メモリを節約する分離技法（再度、Jensenの文献の第３章を参照されたい）を用いることができなくなる。これは、ベイズ・ネットワークが、任意のテストによって網羅されるコンポーネント数に関して、指数関数的な量のメモリを必要とするということである。小規模なモデルであっても、デスクトップ型ＰＣワークステーションのメモリを使い果たす。このアプローチが、組み込みアプリケーションまたは分散アプリケーションにあまり適さないのは明白である。

【００３７】従って、本発明によって適用されるモデルのクラスは、ベイズ・ネットワークのサブクラスとみなすことができる。本発明の診断アルゴリズムは、このサブクラスに関する信念レビジョンに対して有効なアルゴリズムとみなすことが可能である。本発明に関する好結果の診断の正確率が高いのは（後で示すように）、このサブクラスが、実際の診断問題を表すのに十分なほど強力であることを示唆している。さらに、ベイズ・ネット

ワークの自由な形の構成と比べると、本発明によるモデルの相対的な構造特性は、モデルを構築し、支援する場合に有利であると思われる。

【００３８】ベイズ・ネットワークと同様、本発明では、コンポーネントの故障尤度が正確に計算される。ヒューリスティック及びモンテカルロ・シミュレーションでは、近似尤度が計算される。本発明の実行時性能は、実際に良好である。その実行速度は、同じ診断問題について、前述のHP Fault Detectiveとほぼ同じである。

【００３９】ごく簡単に言えば、本発明は、下記の仮定に基づくものである。

１．コンポーネント状態（すなわち、各コンポーネントが良好か、あるいは、不良か）は、確率的に独立している。

２．共用関数状態（すなわち、各共用関数が合格か、あるいは、不合格か）は、コンポーネント状態が与えられると、確率的に独立したものになる。

３．テスト状態（すなわち、各テストが合格、あるいは、不合格か）は、コンポーネント状態及び共用関数状態が与えられると、確率的に独立したものになる。

【００４０】仮定１を、任意の特定の集合をなすコンポーネントが不良であり、他の全てが良好である確率の計算に用いる。

【００４１】仮定２を、ある特定の集合をなすコンポーネントが不良の場合に、任意の特定の集合をなす共用関数が不合格であり、別の特定の集合をなす共用関数が合格する確率の計算に用いる。

【００４２】仮定３を、特定の集合をなすコンポーネントが不良（そして、残りが良好）であり、特定の集合をなす共用関数が不合格になる（そして、残りが合格する）場合に、任意の特定の集合をなすテストが不合格になり、別の特定の集合をなす共用関数が合格する確率の計算に用いる。

【００４３】従って、本発明は、下記の手段を備える。

１．コンポーネントの事前確率と、網羅度と、共用関数網羅度とを指定する手段。

２．どのテストが合格し、どのテストが不合格になったかを指定する手段（テストの中には、実施されなかったため、合格でもなければ、不合格でもないものもあり得る。）

３．どれだけ多くのコンポーネント同時に不良になる可能性があるかを指定する手段。この数をＮと称する。

４．サイズがＮ以下であるコンポーネントの部分集合のそれぞれが、不良コンポーネントである確率を計算する手段。これにより、

・計算が正確になり（わずかな浮動小数点計算の誤差範囲内まで）、

・計算の実施に必要なメモリ量が、入力及び出力の記憶に必要とされるメモリ量よりも、ある一定の量だけ大きいことが望ましいということになる。（ある「一定の量だけ大きい」は、モデル・サイズ及びＮとは関係なく、同じ増大量を表すものとする。）

５．確率を出力する以下のいずれかの手段。

・人間に読み取り可能な形式。

・後続の自動処理のために利用可能なコンピュータ・データ。

【００４４】前記の手段３の代わりに、デフォルト状態の値（おそらく１または２）を組み込むことも可能である。これによって、本発明を利用する場合の柔軟性が低下することになるが、その有用性はあまり損なわれない。

【００４５】本発明によれば、必要になるメモリ量が、モデルの記憶に必要なメモリ量、及び、モデル及び出力サイズに関係なく一定の小さい因子が乗ぜられる出力の記憶に必要なメモリ量より少ない診断エンジンの構成が可能になる。このことは、このような診断エンジンを、組み込み診断エンジンとしての使用に適したものとする。

【００４６】任意の種類のデータ記憶媒体に記憶しておくか、または、前記データ記憶媒体によって別様に供給を受けることが可能であり、任意の適合するデータ処理装置において、あるいは任意の適合するデータ処理装置によって実行することが可能な１つ以上の適合するソフトウェア・プログラムによって、本発明を部分的または全体的に実施することが可能であることは明らかである。

【００４７】

【発明の実施の形態】表１は、コンポーネントとテストと共用関数とのような網羅度ベース・モデルからの項目の概念を示し、網羅度及び共用関数の変動性に関する形式的な定義を下している。

【表１】

15　　16

表記法の要約

Φ	コンポーネントの集合
Ψ	テストの集合
Ω	共用関数の集合
c	コンポーネント
t	テスト
s	共用関数
C	コンポーネントの集合、C⊆Φ
T	テストの集合、T⊆Ψ
S	共用関数の集合、S⊆Ω
π	合格したテスト、π⊆Ψ
φ	不合格のテスト、φ⊆Ψ
sfcov（s，c）	コンポーネントcに関する共用関数sの網羅度
cov（t，c）	コンポーネントcに関するテストtの網羅度
sfused（t）	テストtで用いられる共用関数の集合
sfvar（s）	共用関数sの変動性
sfprob（s）	s∈sfused（t）によるテストtが、sの不合格のために不合格になる確率
k（A）	確率変数集合Aの構成集合
N	同時に不合格になるコンポーネントの想定最大数
M	集合区画における要素数
∋	「のような」
｜A｜	集合Aの基数
A	集合Aの補数

【００４８】単純化のため、「方程式」という用語は、下記の純粋な数学的な方程式だけではなく、この説明において言及される数学的な項についても用いられるものとする。

【００４９】コンポーネントΦは、状態｛良好、不良｝を備えることが可能な確率変数である。このモデルでは、c∈Φの場合の事前確率P（c　bad）が与えられる。コンポーネント故障は、（数１）の方程式に定義のように、独立しているものと仮定される。

【数１】

$$P(C \text{ bad},\ C \subseteq \Phi) = \prod_{c \in C} P(c \text{ bad})$$

【００５０】共用関数は、テスト間の確率的依存性を表＊すために用いられる。共用関数は、いくつかの機能性は、異なるテストによって全く同じやり方で検査されるという事実を表すものとみなすことが可能である。共用関数Ωは、状態｛合格、不合格｝を備える確率変数である。共用関数sは、（数２）の方程式に示すようにコンポーネントΦの状態によって決まり、このモデルでは、共用関数網羅度sfcov（s，c）が与えられる。共用関数は、（数３）の方程式に示されるように、条件付きで互いに独立している。

【数２】

$$P(s \in \Omega \text{ failed} | c \in \Phi \text{ bad}) = \text{sfcov}(s, c)$$

【数３】

$$P(S \subseteq \Omega \text{ fail} \mid C \subseteq \Phi \text{ bad},\ \overline{C} \text{ good}) = \prod_{s \in S} P(s \text{ fail} \mid C \subseteq \Phi \text{ bad},\ \overline{C} \text{ good})$$

【００５１】直観的に、共用関数は、それが網羅する任意のコンポーネントにおいて不良スポットを網羅している場合に不合格になる。正式には、全てのコンポーネントの状態によって決まる共用関数sが不合格になる確率※は、（数４）及び（数５）の方程式によって定義される。

【数４】

$$P(s \in \Omega \text{ fail} | C \subseteq \Phi \text{ bad},\ \overline{C} \text{good}) \ = \ 1 - \prod_{c \in C} (1 - \text{sfcov}(s, c))$$

【数５】

17

$$= 1 - \prod_{\substack{c \in C \\ \mathrm{sfcov}(s,c) \neq 0}} (1 - \mathrm{sfcov}(s,c))$$

【００５２】（数５）の方程式は、ｓｆｃｏｖの疎表現（sparse representation）を繰り返す間に、計算を実施することが可能であることを表している。各共用関数ｓは、０～１の変動性ｓｆｖａｒ（ｓ）を有している。直観的に、共用関数の変動性は、その共用関数が不合格の場合に、その共用関数を用いたテストの不合格が相関する程度を表している。０の変動性は、その共用関数が不合格の場合には、その共用関数を用いた全てのテストが不合格になることを表している。１の変動性は、その共用関数が不合格の場合には、その共用関数を用いたテストが互いに関係なく不合格になることを表している。この場合、共用関数は、モデル化の利器（modeling con*

18

*venience）として用いられている。共用関数の変動性の概念は、次のようにまとめ上げられる。

【００５３】テストΨは、状態｛合格、不合格｝を備えた確率変数である。一般に、テストの一部だけしか実施されない。Πを合格したテストとする。ψが不合格になったテストとする。テストは、コンポーネントと共用関数の両方によって決まる。網羅度Ｐは、（数６）の方程式によって定義されるモデルで示される。テストによって用いられる共用関数ｓｆｕｓｅｄ（ｔ）⊆Ωもモデルで示される。テストは、（数７）の方程式において示される全てのコンポーネント及び共用関数の状態が与えられると、条件付きで互いに独立することになる。

【数６】

$$P(t \in \Psi \text{ failed} | c \in \Phi \text{ bad}) = \mathrm{cov}(t,c)$$

【数７】

$$P(T \subseteq \Psi \text{ fail} \mid C \text{ bad}, \overline{C} \text{ good}, S \text{ fail}, \overline{S} \text{ pass}) = \prod_{t \in \Psi} P(t \text{ fail} \mid C, \overline{C}, S, \overline{S})$$

【００５４】テストで共用関数が用いられない場合、その不合格の確率は、コンポーネントの状態によって決まる。直観的に、不良スポットを網羅している場合には、テストは不合格になる。正式には、テストｔで共用関数が用いられない場合には、全てのコンポーネントの状態※によって決まるｔが不合格になる確率は、（数８）及び（数９）の方程式によって定義されている。（数９）の方程式は、ｃｏｖの疎表現を繰り返すことによって、計算を実施することが可能であることを表している。

【数８】

$$P(t \in \Psi \text{ fail} \mid C \subseteq \Phi \text{ bad}, \overline{C} \text{ good}) = 1 - \prod_{c \in C} (1 - \mathrm{cov}(t,c))$$

【数９】

$$= 1 - \prod_{\substack{c \in C \\ \mathrm{cov}(t,c) \neq 0}} (1 - \mathrm{cov}(t,c))$$

【００５５】テストで共用関数が利用される場合には、共用関数のどれかが不合格になると、テストも不合格になる可能性がある。（数１０）の方程式を仮定する。そ★して、テストがうまくいく条件付き確率は、（数１１）の方程式によって示され、テストが不合格になる条件付き確率は、（数１２）の方程式によって示されるその補数である。

【数１０】

$$\mathrm{sfprob}(t) = 1 - \mathrm{sfvar}(t)/2$$

【数１１】

$$P\left(t \in \Psi \text{ passed} \,\middle|\, C \subseteq \Phi \text{ bad}, \overline{C} \text{ good}, S \subseteq \Omega \text{ failed}, \overline{S} \text{ passed}\right) = \prod_{c \in C} (1 - \mathrm{cov}(t,c)) \prod_{s \in \mathrm{sfused}(t)} (1 - \mathrm{sfprob}(s) P(s \text{ failed} \mid C, \overline{C}, S, \overline{S}))$$

【数１２】

$$P(t \in \Psi \text{ failed} \mid C \subseteq \Phi \text{ bad}, \overline{C} \text{ good}, S \subseteq \Omega \text{ failed}, \overline{S} \text{ passed}) = 1 - \prod_{c \in C} (1 - \mathrm{cov}(t,c)) \prod_{s \in \mathrm{sfused}(t)} (1 - \mathrm{sfprob}(s) P(s \text{ failed} \mid C, \overline{C}, S, \overline{S}))$$

【００５６】確率変数Ωと、Φと、Ψとの３集合の間における全ての確率的依存性が、前述の方程式において示される。さもなければ、確率変数は独立している。従って、確率変数間の依存性は、閉路なし有向グラフ（ＤＡＧ）のソースがコンポーネントΦであり、シンクがテストΨであるベイズ・ネットワークによって表すことが可能である。例えば、ｃｏｖ（ｔ，ｃ）のようなｃｏｖにおける各非ゼロ項目は、コンポーネント・ノードｃから共用関数ノードｓまでのエッジを生じる。各要素ｓ∈ｓｆｕｓｅｄ（ｔ）毎に、共用関数ノードｓからテスト・ノードｔまでのエッジが存在する。

【００５７】上記定義が与えられると、本発明による診断アルゴリズムを生成することが可能になる。このアルゴリズムは、テスト結果が与えられると、コンポーネント構成の事後尤度を計算し、分類するだけのことである。Π⊆Ψが不合格になったテストであると仮定する。

ベイズ・ルールによって、（数１３）の方程式が得られる。

＊【数１３】

$$P(C \text{ bad}, \overline{C} \text{ good} \mid \pi \text{ pass}, \phi \text{ fail}) = \frac{P(\pi \text{ pass}, \phi \text{ fail} \mid C, \overline{C}) P(C, \overline{C})}{P(\pi \text{ pass}, \phi \text{ fail})}$$

【００５８】これらの条件付き確率は、全て、同じ量Ｐ（Π，ψ）によって正規化されることになる。この量は、テスト結果の事前確率であり、計算が困難である。※

※従って、本発明では、（数１４）の尤度を利用する。

【数１４】

$$L(C \text{ bad}, \overline{C} \text{ good} \mid \pi, \phi) = P(\pi \text{ pass}, \phi \text{ fail} \mid C, \overline{C}) P(C, \overline{C}).$$

【００５９】計算すべき唯一の非自明な量は、（数１５）である。共用関数がない場合、これは容易であり、（数１６）の方程式が得られることになるが、ここで、テスト結果が与えられると、合格することになるテストの確率（数１７）は、（数１９）～（数２１）の方程式において示され、テスト結果が与えられると、不合格になるテストの確率は、（数２２）及び（数２３）の方程式において示される。

【数１５】

$$P(\pi, \varphi \mid C, \overline{C})$$

【数１６】

$$P(\pi, \phi \mid C, \overline{C}) = P(\pi \text{ pass} \mid C, \overline{C}) P(\phi \text{ fail} \mid C, \overline{C})$$

【数１７】

$$P(\pi \mid C, \overline{C})$$

【数１８】

$$P(\varphi \mid C, \overline{C})$$

【数１９】

$$P(\pi, \phi \mid C, \overline{C}) = \sum_{(\sigma \text{ failed}, \overline{\sigma} \text{ passed}) \in \kappa(\Omega)}$$

★【数２０】

$$P(\pi \text{ pass} \mid C, \overline{C}) = \prod_{t \in \pi} P(t \text{ pass} \mid C, \overline{C})$$

$$= \prod_{t \in \pi} \prod_{c \in C} P(t \text{ pass} \mid c)$$

【数２１】

$$= \prod_{t \in \pi} \prod_{c \in C} (1 - \text{cov}(t, c))$$

【数２２】

$$P(\phi \text{ fail} \mid C, \overline{C}) = \prod_{t \in \phi} P(t \text{ failed} \mid C, \overline{C})$$

【数２３】

$$= \prod_{t \in \phi} \left(1 - \prod_{c \in C} (1 - \text{cov}(t, c))\right)$$

【００６０】共用関数がある場合、（数２４）の方程式の全確率の法則を利用する。この場合、被加数における第１の因子が、次に、（数２５）の方程式において示されるように、（数１１）及び（数１２）の方程式に従って計算される因子の積になる。

★【数２４】

$$P(\pi, \phi \mid C, \overline{C}, \sigma, \overline{\sigma}) P(\sigma \text{ failed}, \overline{\sigma} \text{ passed} \mid C, \overline{C})$$

【数２５】

$$P\left(\pi, \phi \mid C, \overline{C}, \sigma, \overline{\sigma}\right) = \prod_{t \in \pi} P(t \text{ passed} \mid C, \overline{C}, \sigma, \overline{\sigma}) \prod_{t \in \phi} P(t \text{ failed} \mid C, \overline{C}, \sigma, \overline{\sigma}).$$

【００６１】共用関数の状態の条件付き確率は、（数２６）～（数２８）の方程式において示されるように、（数１６）の方程式のテスト結果の確率と全く同様に計☆算される。

【数２６】

$$P(\sigma \text{ fail}, \overline{\sigma} \text{ pass} \mid C, \overline{C}) = P(\sigma \text{ fail} \mid C, \overline{C}) P(\overline{\sigma} \text{ pass} \mid C, \overline{C})$$

【数２７】

$$P(\sigma \text{ fail} \mid C, \overline{C}) = \prod_{s \in \sigma} \left(1 - \prod_{c \in C} (1 - \text{sfcov}(s, c))\right)$$

【数２８】

$$P(\overline{\sigma} \text{ pass} \mid C, \overline{C}) = \prod_{s \in \overline{\sigma}} \prod_{c \in C} (1 - \text{sfcov}(s, c))$$

【００６２】（計算時間の改善）診断は、コンポーネント及び共用関数の可能性のある状態毎に、（数１４）、（数１６）、（数１９）～（数２４）の方程式を直截的に（straightforward）評価することによって実施可能である。しかし、そのアプローチには、（数２９）の時間がかかるが、これは、明らかに、大部分の実際のアプリケーションにとって許容できるものではない。本発明

によれば、計算時間を短縮する技法を適用することが可能であるが、その最も重要な点は次の通りである。

・診断候補数、すなわち、（数１４）の事後確率方程式の計算対象となるコンポーネント状態である（数３０）の数を減少させること、及び、

・合計に影響しない共用関数のべき集合の状態を排除することによって、（数２４）の方程式の評価に必要な時間を短縮すること。

【数２９】

$$O\left(2^{|\Phi|+|\Omega|}\right)$$

【数３０】

$$(C,\ \overline{C})$$

【００６３】ａ）診断候補数の削減

まず、コンポーネント状態の数を減少させるヒューリスティックについて考察してみることにする。これは、同時に故障するコンポーネントの最大数に関する妥当な仮定を行うことによって実現可能である。本発明では、コンポーネント故障は独立したものと仮定する。従って、故障の事前確率が大きくない限り、複数の故障はまれにしか起こらない。この観測結果は、同時に起こる故障の最大数Ｎを選択し、１≦｜Ｃ｜≦Ｎのとき、それらのＣ⊆Φについてのみ、（数１４）の方程式を計算することを示唆している。これは、本発明で用いるのが望ましい手法である。

【００６４】もう１つの手法は、前述のHP Fault Detectiveによって用いられているものであり、必要な数の故障コンポーネントだけを前提とするオッカムの剃刀（Occam's Razor）に基づくものである。換言すれば、Ｎ＝１を選択し、その尤度を計算する。尤度が非ゼロの場合には中止する。さもなければ、Ｎを１だけ増し、繰り返す。こうして、テスト結果を説明するのに必要な基数（cardinality）を最低にして、１集合の診断が得られる。このアプローチには、２つの危険がある。

１．テスト間に非モデル化依存性が存在する異常な状況の場合、妥当な時間量内で、アルゴリズムを中止することができない。これは、例えば、テスト装置が間違ってセットアップされた場合に発生する可能性がある。

２．ベイズ・アルゴリズムは、診断について、どれほどであろうと可能性があれば、非ゼロの尤度を生じる。ある尤度のしきい値を設定しなければならないが、決定し難い分母（denominator）が、（数１３）の方程式から省略されている場合には、設定が困難である。

【００６５】この手法は、HP Fault Detectiveにはうまく働くが、本発明の場合には、極めて尤度の低い診断候補を見つける可能性があるので、うまく働かない。｜Ｃ｜＝１の場合であっても、本発明では、例えば、10^{-50}といった低尤度の診断、あるいは、10^{-100}といった低尤度の診断でさえ求めることになる。

【００６６】ｂ）活動共用関数

次に、（数２４）の方程式の和を求めるべき集合ｋ（Ω）のサイズを縮小する問題について考察することにする。共用関数は、それに網羅されるコンポーネントの診断についてのみ、及び、実施される少なくとも１つのテストが、共用関数を利用する場合の診断に限って、役割を果たすのは明らかである。従って、（数２４）の方程式は、（数３１）の方程式のはるかに小さいべき集合について和を求めることが可能であるが、ここで、（数３２）は、（数３３）の方程式において定義される暫定活動共用関数を利用する（数３４）の方程式において定義される活動共用関数である。

【数３１】

$$(\sigma\ \text{failed}\ ,\overline{\sigma}\ \text{passed}) \in \kappa\left(\hat{\Omega}\right)$$

【数３２】

$$\hat{\Omega}$$

【数３３】

$$\hat{\Omega} = \left\{ s \in \tilde{\Omega} \mid \exists c \in C \ni \text{sfcov}(s,c) > 0 \right\}$$

【数３４】

$$\tilde{\Omega} = \bigcup_{t \in \pi,\phi} \text{sfused}(t)$$

【００６７】任意の共用関数ｓに対してｋ（Ω）の状態を対にし、ｓが一方において合格し、もう一方において不合格になる点を除いて、各対の要素が、同じになるようにすることができるので、（数３５）に対する制限が、方程式において正当化される。ｓが（Π∪ψ）のどのテストにも用いられない場合、（数３６）の方程式は、その対に関して不変であり、（数３７）の方程式の和によって、その状態に関する他のコンポーネントの確率が得られる。従って、前記対の和を求めることによって、ｓは（数２４）の方程式の目的からはずされる。

【数３５】

$$\kappa(\tilde{\Omega})$$

【数３６】

$$P(\pi,\phi \mid C,\overline{C},\sigma,\overline{\sigma})$$

【数３７】

$$P(\sigma\ \text{failed},\ \overline{\sigma}\ \text{passed} \mid C,\overline{C})$$

【００６８】（数３２）の制限に関して、（数２７）の方程式の考察を行うことにする。共用関数ｓ　∈　σが、任意の推定される故障コンポーネントｃ　∈　Ｃを網羅していない場合、ｓｆｃｏｖ（ｓ，ｃ）は一様にゼロになるが、これは、（数２７）の方程式の最も内側の積がそのｓに関して１になることを意味している。この結果、最も外側の積にゼロの因子が強制的に含められ、（数３８）の方程式が得られる。その結果が、（数２

23

６）の方程式を経て（数２４）の方程式に戻り、全項がゼロになる。従って、こうした共用関数について不合格と仮定する状態項を評価する必要がなくなる。さらにまた、ある状態が、その共用関数を合格と仮定する場合、それによって、ただ単に、（数２８）の方程式の積に「１」が組み込まれるだけである。従って、（数２４）の方程式の和が求められる状態に前記共用関数を含める理由がない。

【数３８】

$$P\left(\sigma \text{ fail} \mid C, \overline{C}\right) = 0$$

【００６９】（数３９）の暫定活動共用関数集合は、実施されたテストだけに依存しているので、診断開始時に１回迅速に計算することが可能である。実施されたテストが、利用可能なテストに比べて少ない場合、これによって、検討を受ける共用関数の数をかなり削減することが可能である。（数３２）の活動共用関数集合は、評価を受ける故障コンポーネントの各組み合わせ毎に、この集合から個別にふるいにかけられる。検討する同時故障の数を制限すると（上記参照）、通常、この集合のサイズは大幅に縮小される。

【数３９】

$$\tilde{\Omega}$$

【００７０】表２には、さまざまなモデルの活動共用関数の数に関するいくつかの例が示されている。表の最初の４列は、モデル名と、モデルにおけるコンポーネントと、テストと、共用関数の数とが示されている。列５には、展開される状態の観測が行われた活動共用関数の最＊

24

＊大数が示されている。その多くの共用関数は、必ず遭遇するとは限らない。列７には、（数２４）の方程式の展開が行われる、べき集合の平均サイズが示されているが、それは、下記の（数４０）の平均である。これは、共用関数の処理に費やされる計算時間の因子である。列６は、列７の底が２の対数であり、活動共用関数の有効「平均」数を示す。データ・セットは、ボイシ（Boise）ディスク・ドライブ・コントローラ・ボード用のものであり、リンクス（Lynx）３は、ＰＣからのボードである。それらに関して観測された数字が、多くの一連の実際のテスト結果について導き出され、有効平均ＳＦ（shared function）数字は、小数第２位までほぼ必ず同じになった。Cofxhfdfは、無線と、測定装置と、それらの間のケーブルとで一杯の小さな構造であるスペクトル・モニタ・システムのモデルである。表の数字は、任意に最初の３０のテストを合格にし、次の３０のテストを不合格にすることによって導き出されたものである。これは、人為的なテスト結果であるが、スペクトル・モニタ・システムの場合、こうした多数のテスト不合格は生じない。その結果は有望である。というのも、５．８５の膨張係数は、2^{203}には程遠いからである。その診断の実行には、プログラムのローディング及び起動時間とモデルへの読み取り時間とを含めて、２００ＭＨｚのペンティアム（登録商標）・プロ・コンピュータで７．９秒の実時間を要する。プログラムはＪａｖａで書かれる。

【表２】

モデル	#コンポーネント	#テスト	#SF	#aSF max	#aSF eff. Abg.	#SF展開
Boise	44	40	6	4	3.48	11.15
Lynx3	53	67	3	3	1.94	3.83
Cofxhfdf	284	184	203	6	2.55	5.85

【数４０】

$$2^{\#\text{active SFs}}$$

【００７１】ｃ）短絡

１００％の網羅度で合格したテストが、想定した不良コンポーネントの疑いを晴らす場合には、（数２５）の方程式の第１の積は、ゼロとすることが可能である。モデルが１００％の網羅度を要求するのは、実際には良くないやり方であり、従って、これを検査する価値はないかもしれない。より関心のある事例は、第２の積の項がゼロという場合である。これは、想定した不良コンポーネントによって、不合格のテストの１つが不合格になる可能性はないということを表している。不必要な処理を回避するため、この状態には検査するだけの価値がある。

【００７２】ｄ）因数分解

（数２４）の方程式の和を計算する場合には、その第１の因子は、その第１の積が（数１１）の方程式によって計算される、（数２５）の方程式に従って展開される。これには、さらに、その和の全ての項に関して不変であり、従って、ループから取り除くことが可能な（数４１）の方程式の因子が含まれている。

【数４１】

$$\prod_{c \in C} (1 - \text{cov}(t, c))$$

【００７３】ｅ）その他

上記スピードアップによって、アルゴリズムの複雑性の次数が減少する。他のプログラミング技法も、必要な処理時間の短縮に役立つ。例えば、不合格になったテストの網羅度は、不合格になったコンポーネントと照合しな

ければならない。これは、コンポーネントとテストと網羅度とを記憶したままであれば、より速くなる。活動共用関数集合をふるいにかける場合と同様、ビットマップを利用して、集合の交差及び合併を計算することが可能である。しかし、活動共用関数集合は、その状態の全てを列挙するため、圧縮表現のままにしておくのが望ましい。可能であれば、実行中における配列の割り当て及び解除を避けるため、配列を事前に割り当てるのが望ましい。

【００７４】（結論）当業者には、以上の詳細な説明及び下記の手続きから明らかなように、本発明に従って構成された診断エンジンが必要とするメモリ量は、モデルを記憶するのに必要なメモリ量、及び、モデル及び出力サイズに関係なく、定数である小さい因子が乗ぜられる出力を記憶するのに必要なメモリ量より少ない。これにより、こうした診断エンジンは、組み込み診断エンジンとして用いるのに適したものになる。

【００７５】少ないメモリ消費を実現する効果は、テスト結果が与えられると、コンポーネント故障の条件付き尤度を正確に計算するために用いられる方程式を利用することによって得られる。上記方程式の番号付与を利用すると、これは、（数２４）の方程式（これは、さらに、（数２５）の方程式を利用し、（数２５）の方程式は、さらに、（数２６）と（数２７）と（数２８）との方程式を利用する）及び（数１）の独立方程式から計算しなければならない値を含む（数１４）の方程式になる。しかし、これらの方程式の内容は、本発明の範囲を逸脱することなく、他の方程式によって表すことも可能であることは明白である。

【００７６】本発明によれば、モデル及び出力の記憶に必要なメモリのほかに、診断の計算において、メモリを必要とすることはほとんどなくてすむことになる。より明確に例示するため、診断の計算にはどんな追加メモリが必要になるか明らかにすることにする。少ないメモリ消費の効果は、その追加メモリを利用する特徴から得られる。

【００７７】これらの（数１４）の方程式及び（数２４）～（数２８）の左側の値を右側から計算すると、下記以外の中間結果の記憶域が不要になる。

・（数２４）の方程式における和を累算するための浮動小数点レジスタまたは記憶場所。

・（数２５）の方程式における積を累算するための浮動小数点レジスタまたは記憶場所。

・（数２７）の方程式における積を累算するための２つの浮動小数点レジスタまたは記憶場所。

・（数２８）の方程式における積を累算するための１つの浮動小数点レジスタまたは記憶場所。

【００７８】すなわち、（数１４）の方程式及び（数２４）～（数２８）の計算に必要とされるのは、最少で４つの浮動小数点レジスタまたは４つの記憶場所ということになる。

【００７９】計算時間を改善するため、活動共用関数（（数３３）の方程式）を適用することが可能である。しかし、こうすると、活動共用関数を計算し、記憶するため、必要なメモリの量が増大する。メモリに記憶しなければならないオブジェクトが２つ存在する。すなわち、暫定活動共用関数と共用関数である。

【００８０】暫定活動共用関数は、一般に、計算の開始前に、通常１回計算される。暫定活動共用関数は、共用関数の部分集合である。暫定活動共用関数を記憶する方法の１つは、整数配列であり、この場合、配列の各整数によって、暫定活動共用関数である共用関数の指標が得られる。従って、暫定活動共用関数は、ｐの整数サイズの記憶場所に記憶することができるが、ここで、ｐは、共用関数の数より少ない暫定活動共用関数の数である。活動共用関数は、診断計算を行う間に変化する。しかし、任意のある時間に存在する活動共用関数の集合は１つだけである。活動共用関数は、暫定活動共用関数の部分集合である。従って、活動共用関数は、共用関数の数と同じ整数の記憶場所に記憶することが可能である。

【００８１】ごく簡単に言えば、これは、少ないメモリ消費の効果が、（数１４）の方程式及び（数２４）～（数２８）のような、統計方程式の直截的で正確な評価から得られるということである。これらの方程式の左側の値の計算には、ほんのわずかな浮動小数点レジスタ／メモリしか必要としない。この評価をより効率よく実施するため、暫定活動共用関数及び活動共用関数の集合を計算することも可能である。これらの集合は、それぞれ、モデルにおける共用関数の数と同じ整数の記憶場所を必要とする。従って、診断の計算に必要な仮記憶場所の数は、共用関数の数と線形に成長する。全必要メモリ量を得るには、モデル及び診断の出力を記憶するためのメモリも追加しなければならない。

【００８２】（説明に役立つ例）本発明の効果は、さらに記述的に説明することも可能である。多数の組み合わせから最良の組み合わせを求める探索方法の中には、主要な変形方法（variant）が２つ存在する。すなわち、深さ優先(depth-fiest)探索方法と、幅優先(breadth-first)探索方法である。より視覚的な想像力に訴える例として、木に生った最も大きい林檎を見つけなければならないものと仮定する。

【００８３】深さ優先探索の場合、最初の枝まで幹を登り、その枝をたどることになる。その枝が分かれている場合には、より大きいほうの小枝をたどりといったように、各時点において、より大きいほうの小枝をたどっていく。最終的には、林檎または葉または細枝の先端に達することになる。それが林檎であれば、その位置が候補名簿に書き留められ、サイズが測られる。次に、その最後の枝の基部まで降りて、代わりの枝を探査する。いつも、候補名簿に書き留められた林檎より大きい林檎を発

見すると、候補名簿は消去され、新たな林檎の位置及びサイズが書き留められる。最終的には、木全体の探査が済むことになり、候補名簿に、２つ以上の林檎の位置及びサイズが記録されることはない。探査した箇所を記録しておかなければならないが、それには、あまりメモリを必要としない。必要なのは、枝の第１層、第３の代替枝、第２層、第２の代替枝等といった形のリストだけである。その木に、重なり合った枝がわずか１０層しかない場合、しなければならないのは、１０項目のリストを作成することだけである。

【００８４】この手続きには、所定の量の時間が必要になるのは明らかである。最も可能性の高いのは、最も大きい林檎が、大きくて、下に垂れ下がった枝の１つに、または、その近くにあるということである。これを利用するため、幅優先探索を実施することになる。枝の第１層のサイズが測定される。幅優先探索は、最大の枝の林檎を探すことによって開始され、枝をたどって林檎が探し求められる。しかし、測定に取りかかっている枝が他の枝より小さいことが、この作業において、それまでに明らかになると、いつでも、現在の位置を書き留めてその別の枝を探査することになる。こうして、常に、既知の以前に調査されたことのない最大の枝を探査することになる。それまでに最大の林檎が記録しておかれる。小さすぎて、そのサイズの林檎を支えられない枝に遭遇した場合は、いつでも、その枝についてもその小枝についてもそれ以上の探査は不要になる。これによって、戻ることが必要になる可能性のある枝の成長の早いリスト（fast-growing list）が構築されるが、いつも最も可能性の高い場所を調査することになるという報いがある。

【００８５】従って、本発明は、深さ優先探索を行うため、必要な記憶域の量が最小限に抑えられる。計算時間を改善するため、本発明では、「木を調べて」、「大枝の大部分が枯れ、葉及び実のない」ことを発見すると、それらの詳細な検討に煩わされることがなくなるので、幅優先探索を適用することが可能である（共用関数の適用に対応する）。いったん、その木の探索が済むと、枯れた枝を避け、オレンジの木から接ぎ木された枝を無視し続ける。

【００８６】（コンピュータ・コードの例）付録の３つの手続きでは、本発明の実施に利用可能なコンピュータ・コード例の概要が言葉で示される。付録（ａ）の第１の手続き診断Ｄｉａｇｎｏｓｅでは、「計算時間の改善」セクションのａ）及びｂ）に解説の速度の改善を利用して、コンポーネント故障の尤度を計算するための方法を提示する。付録（ｂ）の手続きｅｖａｌＤｉａｇｎｏｓｉｓ、及び、付録（ｃ）の手続きｆｉｎｄＳｆＰＥｓｃは、手続きＤｉａｇｎｏｓｅによって利用される。

【００８７】付録（ａ）の手続きＤｉａｇｎｏｓｅ
パラメータ

・モデルと、
・合格したテスト配列と、
・不合格のテスト配列と、によって、
・尤度の降順に分類された可能性のある診断リスト
を得る。

１．整数の配列であるprovisionalActiveSFsを生成する。これは、合格したテスト配列及び不合格のテスト配列におけるテストのどれもが依存する各共用関数の識別番号を昇順に含んでいる。

２．空集合として診断リストを生成する。

３．Ｎ＝（１～考慮すべき同時故障の最大数）に対して、

ａ．Ｎの故障コンポーネントの全ての組み合わせにわたるＣ（コンポーネント集合）に対して、

ｉ．整数の配列であるactiveSFsを生成する。これは、Ｃにおけるどのコンポーネントにも依存するprovisionalActiveSFsの部分集合を含んでいる。

ｉｉ．evalDiagnosisを呼び出すことによって、Ｃが（しかもＣだけが）故障コンポーネントである尤度を評価し、それをＣ及びactiveSFsに与える。

ｉｉｉ．尤度＞０の場合には、Ｃ及びその関連する尤度を含む項目を診断リストに記入する。

４．尤度の降順に診断リストを分類する。

５．診断リストを戻す。

【００８８】付録（ｂ）の手続きｅｖａｌＤｉａｇｎｏｓｉｓ
パラメータ
・モデルと、
・合格したテスト配列と、
・不合格のテスト配列と、
・想定された不良コンポーネントのリストであるＣと、
・Ｃのコンポーネントに依存し、合格したテストまたは不合格のテストによって利用される共用関数の配列であるactiveSFsとによって、
・合格及び不合格のテストのパターンが、Ｃの全てのコンポーネントの故障によって生じ、他のどのコンポーネントの故障によっても生じなかった可能性のある尤度を示す数を得る。

１．整数の記憶場所におけるビット数ｂを呼び出す。activeSFsがｂより多い要素を有する場合には、エラーを知らせる。（望ましい実施態様では、３２ビットの整数を用いるＪａｖａ言語が利用されるので、望ましい実施態様の場合には、ｂ＝３２になる。）

２．Ppriorは、Ｃの全てのコンポーネントが不合格になるが、他の全てが合格する事前確率を計算する（これは、個々の事前確率の積である。）。

３．sfPEscapeは、不良コンポーネントＣが与えられると、共用関数毎に、共用関数が合格する確率を示す項目を備えた数の配列を計算する。（findSfPEscを参照されたい。）

29

４．sumProb＝０をセットする。
５．sfPattern＝（０～２＾（＃activeSfs）－１（sfPatternを小さいビット配列と解釈する。右から数えて、ビットｉが１であれば、ｉ番目の活動共用関数が不合格になる。さもなければ、その共用関数は合格する。これは、実際には、共用関数の数activeSfs［ｉ］である。））に対して、
ａ．PSFPatは、活動共用関数の個別確率を掛け合わせて、このパターンの発生確率を計算する。それらの確率は、
ｉ．ビットｉがsfPatternにおいて１である場合には、１－sfPEscape［activeSFs［ｉ］］
ｉｉ．ビットｉがsfPatternにおいて０である場合には、sfPEscape［activeSFs［ｉ］］
ｂ．不良コンポーネントＣ及びsfPatternによって表示される不合格の共用関数が与えられると、condPofPassedが、全ての合格したテストが合格すべき確率を計算する。
ｃ．不良コンポーネントＣ及びsfPatternによって表示される不合格の共用関数が与えられると、condPofFailedが、全ての不合格のテストが不合格になるべき確率を計算する。
ｄ．sumProbにPSFPat*condPofPassed*condPofFailedを

30

加える。
６．Prior*sumProbを戻す。
【００８９】付録（ｃ）の手続きｆｉｎｄＳｆＰＥｓｃ
パラメータ
・モデルと、
・想定された不良コンポーネントの配列であるＣと、
・活動共用関数の配列であるactiveSFsとによって、
・不良コンポーネントＣが与えられると、共用関数が合格する確率を示す、モデルに存在する共用関数（活動共用関数だけではない）と同じ数の項目を備えた数の配列を得る。
１．sfPEscapeは、各共用関数（活動共用関数だけではなく、共用関数の全て）毎に１つの項目を備えた新しい数値配列である。
２．sfPEscapeの各要素を１にセットする。
３．ｓ＝（ａｃｔｉｖｅＳＦｓにおける各要素）に対して、
ａ．ｃ＝（ｓによって網羅される各コンポーネント）に対して、
ｉ．ｃがＣにあれば、（１－sfcoverage（ｓ，ｃ））を乗算して、sfPEscape［ｓ］を得る。
４．sfPEscapeを戻す。

フロントページの続き

(71)出願人　399117121
395 Page Mill Road Palo Alto, California U. S. A.

(72)発明者　デビッド・アール・スミス
アメリカ合衆国カリフォルニア州サンノゼ，ミリガン・ドライブ　5454